METHOD AND APPARATUS FOR A COMPUTING SYSTEM HAVING AN ACTIVE SLEEP MODE CPU THAT USES THE CACHE OF A NORMAL ACTIVE MODE CPU

FIELD OF THE INVENTION

The field of invention relates generally to computing systems; and, more specifically, to a method and apparatus for a computing system having an active sleep mode.

BACKGROUND

FIG. **1** shows an exemplary depiction of a computing system **100**. According to the computing system design of FIG. **1**, a central processing unit (CPU) **101** (such as a microprocessor) is used to execute instructions that effectively perform the software routines that are executed by the computing system **100**. The computing system also includes a graphics controller (which may also be referred to as a display controller) **104** that provides digital information (e.g., in the form of bytes of digital data or "words" of digital data that are wider than 8 bits) to a display unit **105**.

The display unit **105** is designed to transform the stream of digital information provided by the graphics controller **104** into orchestrated analog signals that, when applied to a display device (such as a liquid crystal display device or a thin film transistor display device), result in the appearance of visual subject matter (e.g., a graphical user interface (GUI)) on the display unit **105**. The graphics controller **104** is typically designed to perform numerically intensive functions (e.g., that are used to display fluid motions on the display device) so as to offload from the CPU **101** the burden of performing these functions.

In the embodiment of FIG. **1**, a memory controller and bridge unit **102** is coupled to both the graphics controller **104** and the CPU **101**. The memory controller and bridge unit **102** may be implemented, for example, with a pair of semiconductor chips (e.g., a memory controller chip; and, a bridge chip) or a single semiconductor chip. The bridge portion of the memory controller and bridge unit **102** effectively acts as a gateway that allows other "I/O components" $\mathbf{107}_1$, through $\mathbf{107}_N$ (e.g., a disk drive, a CD read only memory (ROM), a networking interface, a diskette drive, a card interface, etc.) to store information into (or retrieve information from) the system memory **103**. Typically, the I/O components share a bus **106** (e.g., a PCI bus) to which the bridge portion of the memory controller and bridge unit **102** is also coupled.

The bus **106** provides an efficient mechanism for sending information between the system memory **103** and the I/O components $\mathbf{107}_1$ through $\mathbf{107}_N$ because each I/O component uses common signal wiring from which the bus **106** is constructed. The bridge portion of the memory controller and bridge unit **102** may translate between a pair of buses (e.g., bus **106** and a second bus (not shown in FIG. **1**) that acts as a third input/output port to the memory controller portion of the memory controller and bridge unit **102**); or, may simply provide a third/input output port to the memory controller portion of the memory controller and bridge unit **102**.

The memory controller portion of the memory controller and bridge unit **102** effectively controls the reading and writing signaling activity (e.g., addressing signals) applied to the system memory **103**. Here, as both the CPU **101** and the various I/O components $\mathbf{107}_1$ through $\mathbf{107}_N$ may invoke the services of the system memory **103** (e.g., in the case of the CPU **101**, for reading instructions or reading/writing data; or, in the case of an I/O component, for forwarding data that will be worked upon by the computing system's software), the memory controller portion of the memory controller and bridge unit **102** may effectively arbitrate or otherwise resolve the contention for the system memory's data storage services that may arise between the various I/O components $\mathbf{107}_1$ through $\mathbf{107}_N$ and the CPU **101**. To the extent that the graphics controller **104** invokes use of the system memory **103**, the memory controller portion of the memory controller and bridge unit **102** may also arbitrate its demands as well.

It is important to point out that other computing system embodiments are possible; and, as such, the term computing system, computer and the like are not to be construed as automatically limited to the exemplary architecture that has been depicted in FIG. **1**. Some exemplary alternative computing system embodiments might entail: 1) coupling the graphics controller **104** to the processor **101** rather than the memory controller and bridge unit **102**; 2) not having a graphics controller **104** (e.g., such that the numerically intensive graphical calculations are performed by the CPU **101**); 3) not having an external (off-chip) cache **108** relative to the CPU **101**; etc. Note that the combination of the CPU **101**, memory controller **102** and system memory **103** (and display controller **104** and external cache **108** if they are implemented) may be referred to as the processing core **109** of the computing system **109**.

Mobile computing systems such as laptop computers, notebook computers, handheld devices (e.g., personal digital assistants, cellphones, IEEE 802.11 based devices, etc.) are often battery powered; and, as such, power consumption is a matter of concern. Typically, the less power consumed by a mobile computing system, the longer the life of the battery that powers the computing system. Often, mobile computing systems are built with a "sleep mode" and/or a "hibernation mode". Either of these modes substantially shut down the activity of the computing system so that battery power is conserved.

In "sleep mode" the computing system's "appendages" outside the processing core **109** (e.g., its display unit **105**, one or more I/O components $\mathbf{107}_1$ through $\mathbf{107}_N$) are shut down while its volatile memory within the processing core **19** (e.g., the external cache **108**, the system memory **103**, etc.) is kept awake (e.g., by continuing to clock/refresh and/or otherwise apply power to the cache and the system memory **103**). The CPU **101** may also shut down various internal units so that the processing of application software effectively ceases. Sleep mode allows the system to conserve battery power consumption (because of the shut down of the appendages and internal CPU units) and also allows the computing system to rapidly awake because its volatile memory was never shut down.

In "hibernation mode" the contents of the volatile memory (e.g., cache and system memory) are first stored to non volatile memory (e.g., a disk drive); and then, the entire system is effectively shut down. Here, typically, greater power savings are realized as compared to the sleep mode because the volatile memory units are shut down. However, it takes longer for the system to return from hibernation mode to its original, normal, active state because the "state" of the system software at the time hibernation mode was entered (as represented by the matter that was transferred from volatile to non volatile memory) needs to be "reloaded" back into volatile memory (e.g., by reading the state data from the disk drive and re-storing it back to its original locations in cache and system memory **103**).